Refine Search

Search Results -

Term	Documents
(13 NOT 11).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	22
(L13 NOT L11).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	22

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L14

Refine Search

Recall Text

Clear

Interrupt

Search History

DATE: Friday, February 13, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L14	L13 not l11	22	L14
L13	sign$2 near5 (2$2 or two$2) near5 complement$3 and (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near20 shift$3	39	L13
L12	L11 not l9	16	L12
L11	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near20 shift$3	25	L11
L10	L9 not l6	0	L10
L9	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near15 shift$3 near9 (left or right or most or least)	9	L9
L8	L7 and shift$3	3	L8

L7	L6 not l5	3	L7
L6	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near15 shift$3 near5 (left or right or most or least)	9	L6
L5	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	6	L5
DB=PGPB,USPT; PLUR=YES; OP=OR			
L4	L3	2	L4
DB=USPT; PLUR=YES; OP=OR			
L3	(2$2 or two$2) near5 complement$3 near8 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	2	L3
L2	(2$2 or two$2) near5 complement$3 near5 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	2	L2
L1	(2$2 or two$2) near5 complement$3 and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	18	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 20 of 22 returned.

□ 1. Document ID: US 6594681 B1

Using default format because multiple data bases are involved.

```
L14: Entry 1 of 22                    File: USPT                 Jul 15, 2003

US-PAT-NO: 6594681
DOCUMENT-IDENTIFIER: US 6594681 B1

TITLE: Quotient digit selection logic for floating point division/square root

DATE-ISSUED: July 15, 2003

INVENTOR-INFORMATION:
```

NAME	CITY	STATE	ZIP CODE	COUNTRY
Prabhu; J. Arjun	Palo Alto	CA		

US-CL-CURRENT: 708/504; 708/653

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 2. Document ID: US 6419638 B1

```
L14: Entry 2 of 22                    File: USPT                 Jul 16, 2002

US-PAT-NO: 6419638
DOCUMENT-IDENTIFIER: US 6419638 B1

TITLE: Optical recognition methods for locating eyes
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 3. Document ID: US 6151682 A

```
L14: Entry 3 of 22                    File: USPT                 Nov 21, 2000

US-PAT-NO: 6151682
DOCUMENT-IDENTIFIER: US 6151682 A

TITLE: Digital signal processing circuitry having integrated timing information
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 4. Document ID: US 6095989 A

L14: Entry 4 of 22 File: USPT Aug 1, 2000

US-PAT-NO: 6095989
DOCUMENT-IDENTIFIER: US 6095989 A

TITLE: Optical recognition methods for locating eyes

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 5. Document ID: US 5993051 A

L14: Entry 5 of 22 File: USPT Nov 30, 1999

US-PAT-NO: 5993051
DOCUMENT-IDENTIFIER: US 5993051 A

TITLE: Combined leading one and leading zero anticipator

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 6. Document ID: US 5954789 A

L14: Entry 6 of 22 File: USPT Sep 21, 1999

US-PAT-NO: 5954789
DOCUMENT-IDENTIFIER: US 5954789 A

TITLE: Quotient digit selection logic for floating point division/square root

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 7. Document ID: US 5825681 A

L14: Entry 7 of 22 File: USPT Oct 20, 1998

US-PAT-NO: 5825681
DOCUMENT-IDENTIFIER: US 5825681 A

TITLE: Divider/multiplier circuit having high precision mode

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 8. Document ID: US 5787030 A

L14: Entry 8 of 22 File: USPT Jul 28, 1998

US-PAT-NO: 5787030
DOCUMENT-IDENTIFIER: US 5787030 A

TITLE: Correct and efficient sticky bit calculation for exact floating point divide/square root results

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 9. Document ID: US 5272660 A

L14: Entry 9 of 22 File: USPT Dec 21, 1993

US-PAT-NO: 5272660
DOCUMENT-IDENTIFIER: US 5272660 A
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for performing integer and floating point division using a single SRT divider in a data processor

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 10. Document ID: US 5237525 A

L14: Entry 10 of 22 File: USPT Aug 17, 1993

US-PAT-NO: 5237525
DOCUMENT-IDENTIFIER: US 5237525 A

TITLE: In a data processor an SRT divider having a negative divisor sticky detection circuit

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 11. Document ID: US 5206825 A

L14: Entry 11 of 22 File: USPT Apr 27, 1993

US-PAT-NO: 5206825
DOCUMENT-IDENTIFIER: US 5206825 A

TITLE: Arithmetic processor using signed-digit representation of external operands

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 12. Document ID: US 5153847 A

L14: Entry 12 of 22 File: USPT Oct 6, 1992

US-PAT-NO: 5153847
DOCUMENT-IDENTIFIER: US 5153847 A
**** See image for Certificate of Correction ****

TITLE: Arithmetic processor using signed digit representation of internal operands

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 13. Document ID: US 5012439 A
L14: Entry 13 of 22 File: USPT Apr 30, 1991

US-PAT-NO: 5012439
DOCUMENT-IDENTIFIER: US 5012439 A

TITLE: Method and apparatus for performing division

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 14. Document ID: US 4881193 A
L14: Entry 14 of 22 File: USPT Nov 14, 1989

US-PAT-NO: 4881193
DOCUMENT-IDENTIFIER: US 4881193 A
**** See image for Certificate of Correction ****

TITLE: Rational number operation unit for reduction

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 15. Document ID: US 4602285 A
L14: Entry 15 of 22 File: USPT Jul 22, 1986

US-PAT-NO: 4602285
DOCUMENT-IDENTIFIER: US 4602285 A
**** See image for Certificate of Correction ****

TITLE: System and method for transforming and filtering a video image

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 16. Document ID: US 4597053 A
L14: Entry 16 of 22 File: USPT Jun 24, 1986

US-PAT-NO: 4597053
DOCUMENT-IDENTIFIER: US 4597053 A

TITLE: Two-pass multiplier/accumulator circuit

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 17. Document ID: US 3936828 A

L14: Entry 17 of 22 File: USPT Feb 3, 1976

US-PAT-NO: 3936828
DOCUMENT-IDENTIFIER: US 3936828 A
**** See image for Certificate of Correction ****

TITLE: VLF navigation system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 18. Document ID: US 3621218 A

L14: Entry 18 of 22 File: USOC Nov 16, 1971

US-PAT-NO: 3621218
DOCUMENT-IDENTIFIER: US 3621218 A

TITLE: HIGH-SPEED DIVIDER UTILIZING CARRY SAVE ADDITIONS

DATE-ISSUED: November 16, 1971

INVENTOR-NAME: NISHIMOTO TETSUNORI

US-CL-CURRENT: 708/655

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 19. Document ID: US 3504167 A

L14: Entry 19 of 22 File: USOC Mar 31, 1970

US-PAT-NO: 3504167
DOCUMENT-IDENTIFIER: US 3504167 A

TITLE: CARRY SELECT DIVIDE DECODE

DATE-ISSUED: March 31, 1970

INVENTOR-NAME: KURTZ CLARK

US-CL-CURRENT: 708/490; 708/504, 708/656

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 20. Document ID: US 3484592 A

L14: Entry 20 of 22 File: USOC Dec 16, 1969

US-PAT-NO: 3484592
DOCUMENT-IDENTIFIER: US 3484592 A

TITLE: UNIFORM SHIFT DIVISION

DATE-ISSUED: December 16, 1969

INVENTOR-NAME: SPCNCER DANA R; TIMM JOE F ; MACSORLEY OLIN L

US-CL-CURRENT: 708/656

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
(13 NOT 11).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	22
(L13 NOT L11).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	22

Display Format: - Change Format

Previous Page Next Page Go to Doc#

First Hit Fwd Refs

□ Generate Collection Print

L14: Entry 16 of 22 File: USPT Jun 24, 1986

DOCUMENT-IDENTIFIER: US 4597053 A
TITLE: Two-pass multiplier/accumulator circuit

Abstract Text (1):
A two-pass Multiplier/Accumulator Circuit is provided which performs various arithmetic operations on operands contained within an X Register 10 (FIG. 1) and a Y Register 20 and places the result in an Accumulator Register 40. The arithmetic operations are carried out by passing the product of the operands successively through an array of adders in the Adder unit 34. Each adder adds an appropriate multiple of the contents of the X Register to the Accumulator 40 or to the output of the previous adder. The multiples are selected according to the contents of the Y Register.

Brief Summary Text (2):
This invention relates generally to digital logic circuits, and, in particular, to a digital two-pass multiplier/accumulator circuit.

Brief Summary Text (5):
In the digital arithmetic technology it is a known problem that digital multiplication and division require complex circuitry and/or complex routines to execute. In the manufacture of high capability, high quality electronic equipment, it is often necessary to provide high speed digital multiplication and divide capability. For such equipment to be economically competitive demands that the manufacturer keep the component costs relatively low. Thus there is a pressing need for a relatively low cost circuit for performing high speed arithmetic operations, including multiplication and division.

Brief Summary Text (8):
Accordingly, it is an object of the present invention to provide an improved Two-Pass Multiplier/Accumulator Circuit.

Brief Summary Text (9):
It is also an object of the present invention to provide a Two-Pass Multiplier/Accumulator Circuit which performs its operations in a pipelined manner to achieve high speed operation.

Brief Summary Text (10):
It is a further object of the present invention to provide a Two-Pass Multiplier/Accumulator Circuit which is capable of determining the maximum or minimum value in a sequence of digital numbers.

Drawing Description Text (3):
FIG. 1 shows a block diagram illustrating a preferred embodiment of a Two-Pass Multiplier/Accumulator Circuit of the present invention.

Drawing Description Text (4):
FIG. 2A shows a timing diagram illustrating the clock timing for the Synchronous Mode (Minimum Cycle Time) of the Multiplier/Accumulator of the present invention.

Drawing Description Text (5):

FIG. 2B shows a timing diagram illustrating the clock timing for the Synchronous Mode (Extended Cycle) of the Multiplier/Accumulator of the present invention.

Detailed Description Text (2):
The Multiplier-Accumulator-RAM circuit (MAR) contains a 16-bit Multiplier/Accumulator (M/A) 1 and a 256-word by 16-bit Random-Access-Memory (RAM) 50 with control circuitry to execute an extensive set of arithmetic and logic functions on data contained in the on-chip RAM or in external devices, as well as a large variety of data transfers between the internal parts of the chip (M/A and RAM) and other external devices. A functional block diagram of the MAR is shown in FIG. 1. This diagram is functional only; and the actual implementation may vary.

Detailed Description Text (5):
MULTIPLIER/ACCUMULATOR/RAM COMPONENTS

Detailed Description Text (19):
Multiply Group

Detailed Description Text (20):
Delayed Multiply Group

Detailed Description Text (35):
Operation of Digital Multiplier/Accumulator

Detailed Description Text (36):
Operation of Pipelined Multiplier

Detailed Description Text (38):
Multiplier/Accumulator/RAM Components

Detailed Description Text (39):
The Multiplier/Accumulator portion 1 of the circuit forms the arithmetic product of two 16-bit numbers contained in registers X and Y, and adds or subtracts this product to or from a 40-bit Accumulator 40 (AX, AH, AL). The M/A can also perform the logical operations AND and EXCLUSIVE OR on the contents of X and AH or AL, and it can shift the entire 40-bit Accumulator contents left or right for normalization, or for multiplication or division by 2. Except for the logical operation, all data are treated as signed numbers using two's-complement notation. A software-controlled mode allows the data to be treated as integers or fractions. The M/A is controlled by the F(0-4) function inputs when enabled by the proper C(0-4) control inputs.

Detailed Description Text (43):
The 16-bit Y Register 20 holds the second operand in Sign Multiply and all ordinary Multiply operations.

Detailed Description Text (47):
The 5-bit input F(0-4) to Function Decode 70 provides 32 M/A functions, including multiply with or without accumulation, add, subtract, compare, absolute value, X.sign of Y, negate, divide, maximum, and minimum.

Detailed Description Text (51):
During one cycle of operation, the MAR decodes the Control and Function inputs, and if so directed, performs a data transfer and a function. In general, the C input controls the data transfer between the RAM, M/A, and DB, and the F input specifies the register in the M/A which is to be read or loaded and the function to be performed in the M/A 1. The function may consist of setting modes or setting or starting an operation. Operations take place during the cycle or cycles following the execution of the function, operating on the data in X, Y, and A, and usually placing the result in A. Except for the logical operations (AND and EOR), all data

are treated as fixed point, two's-complement signed numbers, in either integer or fraction representation. The integer or fraction interpretation is controlled by a mode bit. The most-significant bit of X, Y, or AX indicates the sign of x, Y, or the entire 40-bit Accumulator. The bit-weighting of all registers is shown in Table 1 for integer and fraction representation.

Detailed Description Text (52):
The X and Y registers are fully buffered so that additional data transfers and functions may be performed while an operation is in progress in a "pipeline" fashion. For example, in doing a multiply-accumulate of the form, A=X1.multidot.Y1+X2.multidot.X2+- - - , two operands can be loaded, while the previous operands are being operated upon. This is described in greater detail below in the section below entitled "Pipelined Multiplier".

Detailed Description Text (58):
INTEGER/FRACTION is set by the Load Mode Register function and specifies whether data are to be operated on as integers or fractions. In general, results of multiplications operations are shifted left one bit in FRAC mode, and single-word operations use AH in FRAC mode or AL in INTG mode.

Detailed Description Text (64):
ACCUMULATE is set by the Load Mode Register function. ACC affects only the Absolute Value and Multiply by sign of Y operations. When ACC=0, these operations clear the Accumulator and place the new results into it. When ACC=1, these operations add the new results to the previous Accumulator contents.

Detailed Description Text (76):
In addition to the data transfers stated above, if a Delayed Multiply or a Delayed Multiply-Round operation is pending, it will be started as follows:

Detailed Description Text (82):
In addition to the data transfers, if a Delayed Multiply-Accumulate operation is pending, it will be started as follows:

Detailed Description Text (91):
the mode change will be effective for the operation that follows DIF (MPY in this example). Note, however, that if the LDMR is delayed, such as by an interrupt, the multiply will occur before the LDMR is executed, thus giving different results if the FRAC/INTG mode is changed.

Detailed Description Text (103):
Multiply Group

Detailed Description Text (104):
MULTIPLY

Detailed Description Text (106):
MULTIPLY-ROUND

Detailed Description Text (107):
The product is formed as in MULTIPLY, but in FRAC mode it is added to the quantity 2.sup.-16. The resulting value in AH is X*Y (or -X*Y) rounded to a one-word (16-bit) value. That is, if the most-significant bit in AL due to a MULTIPLY operation in FRAC mode would have been a 1, the MULTIPLY-ROUND operation will produce a value in AH which is rounded to the next higher number than would have been produced by the MULTIPLY. In INTG mode, MULTIPLY-ROUND is the same as MULTIPLY. This operation takes two cycles.

Detailed Description Text (108):
MULTIPLY-ACCUMULATE

Detailed Description Text (109):
The product is formed as in MULTIPLY but it is added to the previous contents of the Accumulator. In FRAC mode, the least-significant bit of AL is ignored, and is forced to 0 in the result. A full 40-bit accumulation is performed. If an overflow of AX occurs, the VX latch will be set to 1.

Detailed Description Text (110):
Delayed Multiply Group

Detailed Description Text (111):
DELAYED MULTIPLY--DELAYED MULTIPLY-ROUND

Detailed Description Text (112):
These operations produce the same results as their counterparts. However, they are not started following the LDX function, but instead following a subsequent RDA function. This permits loading two operands while a previous MULTIPLY operation is in progress, then reading the previous result before starting the new operation. The function that starts these delayed operations is determined by the mode as follows:

Detailed Description Text (116):
DELAYED MULTIPLY-ACCUMULATE

Detailed Description Text (123):
ADD and SUBTRACT are exactly equivalent to the MULTIPLY-ACCUMULATE operation with Y=1 or -1, but are independent of the PLUS/MINUS mode. However, the contents of Y are ignored and are not affected. The value in X (integer or fraction) is added to or subtracted from the 40-bit Accumulator. In FRAC mode, the least-significant bit of AL is ignored, and is forced to 0 in the result. If an overflow of AX occurs, the VX latch will be set. These operations take two cycles in INTG mode, one cycle in FRAC mode.

Detailed Description Text (142):
These operations multiply the contents of X by +1 or -1 and place the result in the Accumulator. If the accumulate mode is set (ACC=1), ABS and MSY add the result to the previous Accumulator contents. The PLUS/MINUS mode affects only the MSY operation.

Detailed Description Text (145):
Multiply by Sign of Y (MSY)--The contents of X are multiplied by +1 or -1 according to the value in Y and the PLUS/MINUS mode as follows:

Detailed Description Text (169):
The DIF operation is executed during the cycle immediately following the LDX function that specifies one of the above operations, including the delayed multiplies. Except for the delayed multiplies, the normal function occurs immediately following DIF. The delayed multiplies are activated in the same manner independent of the DIF operation. At the time the delayed operation is activated, the DIF will have been completed.

Detailed Description Text (195):
Timing of the MAR is controlled by two clock signals supplied to the circuit, the high-speed clock (HCLK) and the cycle clock (CCLK). There are two timing modes determined by the AMODE input, synchronous mode (AMODE=low) and asynchronous mode (AMODE=high). HCLK is a continuous clock which must be supplied regardless of the mode. It runs at a multiple of the cycle frequency and is used to provide internal timing signals. CCLK is used to synchronize the internal timing to external events when necessary, and to initiate asynchronous internal functions. Data Transfer Acknowledge (DTA) is an output used in the asynchronous mode for handshaking with a

host computer.

Detailed Description Text (200):
The minimum period of CCLK is three full cycles of HCLK, with a nominal 50% duty cycle. To extend the cycle, e.g., to transfer data to or from a slower device connected to the data bus, the low (active) phase of CCLK can be extended in multiples of one full cycle of HCLK. The start of the cycle may be delayed by extending the high phase of CCLK also in multiples of one full cycle of HCLK.

Detailed Description Text (217):
Operation of Two-Pass Digital Multiplier/Accumulator Circuit

Detailed Description Text (219):
The M/A multiplies the contents of the two operand data registers, X and Y, and adds the result (product) to the Accumulator A. Optionally, the Accumulator may be cleared to zero at the start of the operation so that the product of X and Y is obtained without accumulating the previous contents of A.

Detailed Description Text (220):
The multiplication and addition to A are accomplished by passing the Accumulator contents twice through the array of Adders. The array comprises Adders 101-104 (FIG. 3) in cascade. Each Adder adds an appropriate multiple of the contents of the X Register to the accumulator or to the output of the previous Adder. The multiples are selected according to the contents of the Y Register. Eight bits (Y') of Y are selected during each pass thus producing a 24-bit result. During pass 1, the low-order 8 bits of Y are selected and the product is added to the low-order 24 bits of A. During pass 2, the high-order 8 bits of Y are selected and the product is added to the high order 24 bits of A.

Detailed Description Text (224):
These multiplex of X are all powers of two and thus require only a left shift and/or inversion of X or forcing of zeros as an input to each Adder.

Detailed Description Text (225):
All operands and results are considered as two's-complement signed numbers. Therefore on pass 2, Y' may be negative whereas on pass 1 it is always considered positive. The Z values are determined by adding the binary value 01010101 to Y' and decoding as per Table 6. The powers of 4 (4, 16, 64) are obtained by shifting the contents of X left two bits at the input of each successive adder stage as shown in Table 6.

Detailed Description Text (226):
Since left-shifting of X produces zeros in the low-order bit positions, Adder stages are not required in the low-order positions for the higher multiples of X, and they are not implemented. The high-order bits of the Adders (beyond 16 bits) are required only to handle the extended Sign of the selected multiple of X, and a possible ripple Carry. Two additional bits (for a total of 18 bits per Adder) can take care of this Sign and Carry with a circuit which provides inputs to the two high-order Adder positions.

Detailed Description Text (231):
The use of a 2-bit-at-a-time algorithm in a parallel multiplier eliminates one-half of the required Adders at the expense of providing a like number of data selectors. In an NMOS circuit, data selectors are made of multiple transmission gates, occupy minimal space, and consume no DC power (except for drivers). The resulting array is therefore considerably smaller and consumes less power than if a conventional parallel array were used.

Detailed Description Text (232):
The number of Adders and data selectors is halved again by making two passes

through the array to complete the multiply. This two-cycle multiple matches very well with the two cycles required to load two operands via the single 16-bit Data Bus 15. As explained below, buffers are provided for the X and Y Registers so that new operands may be loaded while the previously-initiated multiplication is in progress. The penalty for two-cycle multiplication is additional timing logic required, which is much less than the logic saved by halving the Adder array. Three Carry bits must also be saved between passes, but again the required logic is minimal.

Detailed Description Text (233):
Operation of Pipelined Multiplier

Detailed Description Text (235):
The two operand buffer Registers X and Y permit two new operands to be loaded during the 2 cycles in which a multiply operation is taking place on the two operands which were loaded previously. A Function input specifies the register that data is to be loaded or read from during each cycle, and it also specifies, in the case of data to be loaded, the operation to be performed on that data during subsequent cycles. The Control and Timing circuitry implement the loaading, reading, and multiplying.

Detailed Description Text (242):
Load X-Multiply

Detailed Description Text (243):
Load X-Multiply-Accumulate

Detailed Description Text (244):
The Load X-Multiply and Load X-Multiply-Accumulate functions cause the product of X and Y to be formed and added to the Accumulator. However, Load X-Multiply first causes the Accumulator to be cleared to zero.

Detailed Description Text (245):
These functions permit a pipelined multiply-accumulate as follows for the function: Sum (r.sub.i .multidot.s.sub.i) i=1 to n:

Detailed Description Text (247):
For a sequence of non-accumulating multiples of the form:

Detailed Description Text (248):
using only the functions described, six cycles are required per multiply, two of which are wait cycles. In a typical application, only 16-bits of precision are required in the result, so only one READ A needs to be performed, and 5 cycles are sufficient per multiply, two of which are WAITs.

Detailed Description Text (249):
To improve the efficiency of this type of calculation, another function is added: LOAD X-DELAYED MULTIPLY. This function loads data into the X Register and establishes the operation to be performed, but it does not initiate the multiply operation. The multiply operation is automatically initiated following a subsequent READ A. This function permits two operands to be loaded while a previous multiplication is in progress, but permits the previous result to be read before the new operation is performed as shown below.

Detailed Description Text (250):
The total number of cycles is 3n+2 whereas 5n cycles would be required without the DELAYED-MULTIPLY function.

Detailed Description Text (252):
This function is LOADX-DELAYED MULTIPLY-ACCUMULATE. In this case, the operation is

initiated by a LOAD A function rather than a READ A.

Detailed Description Text (254):
The total number of cycles is 4n+2 versus 6n without the DELAYED MULTIPLY-ACCUMULATE function.

Detailed Description Text (276):
It will be apparent to those skilled in the art that the disclosed Two-Pass Multiplier/Accumulator Circuit may be modified in numerous ways and may assume many embodiments other than the preferred form specifically set out and described above.

Detailed Description Text (277):
For example, the Multiplier/Accumulator 1 can be implemented independently of the RAM 50.

Detailed Description Paragraph Table (5):
ADD, SUB all multiples
if .vertline.Y.vertline. .ltoreq. 127 INTG mode only ABS, MSY with ACC = 1

Detailed Description Paragraph Table (14):
Single Multiply: LDYP Y1 LDSMPY X1 NOOP Multiply operation in progress NOOP RDAH X1*Y1 (MS) Result (Most-significant half) RDAL S1*Y1 (LS) Result (least-significant half) Logical AND (INTG mode): LDAL A1 LDXAND X1 NOOP AND operation in progress RDAL A1 .multidot. X1 Result Multiply-Round and ADD (FRAC mode): LDYM Y1 LDXMPR X1 NOOP Multiply operation in progress LDXADD X2 NOOP Add operation in progress RDAH -(X *Y)+Y2 (Rounded to single word) Square of Difference (INTG mode): LDYDIF Y1 LDXMPY X1 NOOP Difference operation in progress NOOP Multiply operation in progress NOOP RDAL (Y1/2 - X1/2).sup.2 (Least significant part) Difference with third operand (FRAC mode): LDYDIF Y1 LDXDMP X1 LDXMPY X2 Difference operation (Difference .fwdarw. Y only) NOOP Multiply operation in progress NOOP RDAH (Y1/2--X1/2)*2 (most-significant part) Multiply-Accumulate: LDYP Y1 LDXMPY X1 LDYP Y2 First multiply in progress LDXMAC Second operand pair loaded . Second multiply in progress . LDYP Yn LDXMAC Xn NOOP Last multiply in progress NOOP RDAL RDAH Result = (X1*Y1) + (X2*Y2) + -- + (Xn*Yn) RDAX Maximum Value (FRAC mode): LDAH X0 LDXMAX X1 NOOP First maximum operation in progress LDXMAX X2 NOOP LDXMAX X3 Second maximum operation in progress . . . LDXMAX Xn NOOP Last maximum operation in progress NOOP RDAH X (Max) Result = maximum of (X0, X1, --,Xn) RDAL i Index: X (MAX) = Xi Array multiplication (delayed multiply) (FRAC mode): LDYP Y1 LDXMPR X1 LDYP Y2 First multiply in progress LDXDMR X2 Load second operand pair RDAH .circle.1 X1*Y1 Read first result LDYP Y3 Second multiply in progress LDXDMR X3 Load third operand pair RDAH .circle.1 X2*Y2 . . . LDYP Yn Multiply in progress LDXDMR Xm Load last operand pair RDAH .circle.1 Xn-1*Yn-1 Read result NOOP Last multiply in progress NOOP RDAH .circle.1 Xn*Yn Read last result .circle.1 If double-word result is desired, read AL first; RDAH starts delayed operation in FRAC mode. Array multiply and Add (Z'.sub.i = X.sub.i *Y.sub.i + Z.sub.i)(INTG mode): LDYP Y1 Load first operand pair LDXDMA X1 LDAL .circle.1 Z1 Load first update value LDYP Y2 First multiply-accumulate operation in progress LDXDMA X2 Load second operand pair RDAL Z'1 Read first result LDAL .circle.1 Z2 Load second update value LDYP Y3 Second multiply-accumulate operation in progress LDXDMA X3 RDAL Z'2 Read second result . . . LDAL .circle.1 Zn NOOP Last multiply-accumulate operation in progress NOOP RDAL Z'n Read last result .circle.1 If double-precision operation is desired, load AH first; LDAL starts DMA operation in INTG mode. Multiply and scale result (INTG mode): LDYP Y1 LDXMPY X1 NOOP Multiply operation in progress NOOP ASR Shift function ASR Shift function RDAL (X1*Y1)/4 Result Divide (Fractional data, FRAC Mode): LDAH A1 (Most-significant) LDAL A1 (Least-significant) LDXSUB X1 LDXDIV X1 Subtract operation in progress NOOP NOOP NOOP NOOP NOOP NOOP NOOP NOOP Divide operation in progress NOOP (16 cycles) NOOP NOOP NOOP NOOP NOOP NOOP NOOP RDAL Quotient=(A1/X1) LDXADD X1 NOOP Add operation in

progress RDAH Remainder Divide (Integer data, FRAC mode): LDAH A1 (Most significant) LDAL A1 (Least significant) ASL Pre-shift divisor LDXSUB X1 LDXDIV X1 Subtract operation in progress NOOP NOOP NOOP NOOP NOOP NOOP NOOP Divide operation in progress NOOP (16 cycles) NOOP NOOP NOOP NOOP NOOP NOOP NOOP NOOP RDAL Quotient= (A1/X1) LDXADD NOOP Add operation in progress ASR RDAH Remainder

Detailed Description Paragraph Table (17):

Function Data Operation in Progress LOAD Y r.sub.1 None LOAD X-MULTIPLY s.sub.1 LOAD Y r.sub.2 r.sub.1 .multidot. s.sub.1 LOAD X-MULT-ACC s.sub.2 LOAD Y r.sub.3 A + r.sub.2 .multidot. s.sub.2 LOAD X-MULT-ACC s.sub.3 . . . . . . LOAD Y r.sub.n A + r.sub.n-1 .multidot. s.sub.n-1 LOAD X-MULT-ACC s.sub.n WAIT -- A + r.sub.n .multidot. s.sub.n WAIT -- READ AH (r .multidot. s) READ AL

Detailed Description Paragraph Table (18):

Function Data Operation in Progress LOAD Y r.sub.1 none LOAD X-MULTIPLY s.sub.1 LOAD Y r.sub.2 r.sub.1 .multidot. s.sub.1 LOAD X-DEL.MULT. s.sub.2 READ A t.sub.1 LOAD Y r.sub.3 r.sub.2 .multidot. s.sub.2 LOAD X-DEL.MULT. s.sub.3 READ A t.sub.2 . . LOAD Y r.sub.n r.sub.n-1 .multidot. s.sub.n-1 LOAD X-DEL. MULT. s.sub.n READ A t.sub.n-1 WAIT -- r.sub.n .multidot. s.sub.n WAIT -- READ A t.sub.n

Detailed Description Paragraph Table (23):

TABLE 2

M/A FUNCTIONS 43210 DIF MNEMONIC DATA TRANSFER FUNCTION/OPERATION CONTROL 00000 RDAL AL .fwdarw. B If INTG mode: start DELAYED 00001 RDAH AH .fwdarw. B If FRAC mode: MULTIPLY or 00010 RDALL AL (LIM) .fwdarw. B .circle.1 If INTG mode: MULTIPLY-ROUND 00011 RDAHL AH (LIM) .fwdarw. B .circle.1 If FRAC mode: 00100 RDAX AX .fwdarw. B None 00101 RDST ST, MR .fwdarw. B None 00110 ASL A (L1) .fwdarw. A Shift A (40 bits) left with 0 fill 00111 ASR A (R1) .fwdarw. A or right with sign extension, 1 bit 01000 * LDXMPY B .fwdarw. X Start MULTIPLY 01001 * LDXMPR B .fwdarw. X Start MULTIPLY-ROUND 01010 * LDXMAC B .fwdarw. X Start MULTIPLY-ACCUMULATE 01011 * LDXABS B .fwdarw. X Start ABSOLUTE 01100 LDXADD B .fwdarw. X Start ADD 01101 LDXSUB B .fwdarw. X Start SUBTRACT 01110 LDAL B .fwdarw. AL .circle.2 If INTG mode: start DELAYED 01111 LDAH B .fwdarw. AH .circle.3 If FRAC mode: MULTIPLY-ACCUMULATE 10000 LDYPM B .fwdarw. Y Set PLUS/MINUS mode (MREV = 0/1) 10001 LDYDIF B .fwdarw. Y Set DIF 10010 LDYP B .fwdarw. Y Set PLUS mode 10011 LDYM B .fwdarw. Y Set MINUS mode 10100 * LDXMAX B .fwdarw. X Start MAXIMUM 10101 * LDXMIN B .fwdarw. X Start MINIMUM 10110 LDXAND B .fwdarw. X Start LOGICAL AND 10111 LDMR B .fwdarw. MR Set/Clear modes 11000 * LDXDMP B .fwdarw. X Set MULTIPLY-DELAYED 11001 * LDXDMR B .fwdarw. X Set MULTIPLY-ROUND DELAYED 11010 * LDXDMA B .fwdarw. X Set MULTIPLY-ACCUMULATE DELAYED 11011 LDXMSY B .fwdarw. X Start SIGN MULTIPLY 11100 * LDXCMP B .fwdarw. X Start COMPARE 11101 * LDXDIV B .fwdarw. X Start DIVIDE 11110 LDXEOR B .fwdarw. X Start LOGICAL EXCLUSIVE OR 11111 LDXNEG B .fwdarw. X Start NEGATE

*Start DIFFERENCE mode if pending. See infra for details of Read A Limited. .circle.2 If FIRST LOAD, propagate sign into AH & AX, then clear FIRST LOAD. .circle.3 Propagate sign into AX: if FIRST LOAD: 0 .fwdarw. AL, then clear FIRST LOAD. All LDX functions set FIRST LOAD and FIRST READ.

Detailed Description Paragraph Table (25):

TABLE 4

M/A OPERATIONS MATHEMATICAL # CYCLES OPERATION MNEMONIC CONDITION DESCRIPTION FRAC INTG

DIFFERENCE DIF (Y/2 - X/2) .fwdarw. Y and 1 1 MULTIPLY MPY, DMP .+-. X * Y .fwdarw. A 2 2 MULTIPLY-ROUND MPR, DMR FRAC 2.sup.-16 .+-. X * Y .fwdarw. A 2 -- INTG .+-. X

* Y .fwdarw. A -- 2 MULTIPLY- MAC, DMA A .+-. X * Y .fwdarw. A 2 2 ACCUMULATE ADD ADD A + X .fwdarw. A 1 2 SUBTRACT SUB A - X .fwdarw. A 1 2 COMPARE CMP MAG = 0 Am - X ; sign .fwdarw. S 1 1 MAG = 1 .vertline.Am.vertline. - .vertline.X.vertline . ; sign .fwdarw. S 1 1 AX + 1 .fwdarw. AX: [COMPARE] MAXIMUM MAX 2 2 if S = 1: X .fwdarw. Am, AX .fwdarw. An AX + 1 .fwdarw. AX: [COMPARE] MINIMUM MIN 2 2 if S = 0: X .fwdarw. Am, AX .fwdarw. An NEGATE NEG -X .fwdarw. A 1 1 ABSOLUTE ABS ACC = 0 .vertline.X.vertline. .fwdarw. A 1 1 ACC = 1 A + .vertline.X.vertline. .fwdarw. A 1 2 SIGN MULTIPLY MSY ACC = 0 .+-.X * sign(Y) .fwdarw. A 1 1 ACC = 1 A .+-.X * sign (Y) .fwdarw. A 1 2 LOGICAL AND AND X .multidot. Am .fwdarw. Am; 0 .fwdarw. An & AX 1 1 EXCLUSIVE OR EOR X .sym. Am .fwdarw. Am; 0 .fwdarw. An & 1X 1 AH (MSB) .fwdarw. M DIVIDE DIV FRAC 2 * A .+-. X + --M * 2.sup.-31 .fwdarw. 16 -- (16 times) INTG Undefined -- 16

.+-.:

Except DIVIDE: + in PLUS mode, - in MINUS mode. DIVIDE: + if M = 1, - if M = 0 Am: AH in FRAC mode, AL in INTG mode An: AL in FRAC mode, AH in INTG mode

CLAIMS:

1. A digital multiplier/accumulator circuit for generating the product of first and second binary numbers, said circuit comprising:

an M-bit multiplicand register for storing said first binary number, said first binary number having M or fewer bits;

an N-bit multiplier register for storing said second binary number, said second binary number having N or fewer bits, said multiplier register being divided into first and second equal portions;

control logic for controlling the operation of said multiplier circuit, said control logic generating at least one control signal in response to one of a plurality of commands applied thereto;

an array of P multiplier decoders, where P=N/4, for decoding a multiplier operand which is stored in either said first portion or said second portion of said multiplier register, and for generating either a first plurality of multiplier decoder outputs or a second plurality of multiplier decoder outputs in response to first and second control signals, respectively, generated successively by said control logic;

the first of said decoders decoding the 2 least significant bits of said multiplier operand and generating 1 of 4 possible first decoder outputs representing multiplication of said multiplicand by the factors 0, +1, -1, or 2, respectively;

a second of said decoders decoding the next 2 higher significant bits of said multiplier operand, and generating 1 of 4 possible second decoder outputs representing multiplication of said multiplicand by the factors 0, +2, -2, or 4, respectively;

each successive decoder decoding the next 2 higher significant bits of said multiplier operand, with the Pth of said decoders decoding the 2 most significant bits of said multiplier operand and generating 1 of 4 possible Pth decoder outputs representing multiplication of said multiplicand by the factors 0, +2.sup.2(P-1), or -2.sup.2(P-1), or 2.sup.2P-1, respectively, where P represents the Pth decoder;

an accumulator register having 2(M+P) bit positions for storing results of the operations of said digital multiplier/accumulator circuit;

an array of P full adders, each being (M+2) bits in length,

said control logic being responsive to said first control signal for causing the least significant M stages of said first adder to be responsive both to the contents of corresponding bits of said multiplicand register and to the output of said first multiplier decoder, and to generate a partial sum equal to the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the two least significant bit positions of said accumulator;

the least significant M stages of each successive adder being responsive to the contents of corresponding bits of said multiplicand register, to the M most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the contents of the M most significant stages of the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire (M+2) bits of the Pth partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the (P-1)th partial sum;

said control logic being responsive to said second control signal for conducting the (M+1)th through (2M+2)th bits of said accumulator to the 1st through (M+2)th stages, respectively, of said first adder and enabling the least significant M stages of said first adder to be responsive to the contents of corresponding bits of said multiplicand register, to the contents of said (M+1)th through (2M)th bits of said accumulator, and to said first multiplier decoder generating one of said second plurality of multiplier decoder outputs, and to thereby generate a patrial sum equal to the sum of said (M+1)th through (2M)th bits of said accumulator and the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the (M+1)th and (M+2)th bit positions of said accumulator;

the (M+1)th and (M+2)th stages of said first adder being responsive to the contents of the (2M+1)th and (2M+2)th bits, respectively, of said accumulator and generating a partial sum;

the two most significant stages of the Pth successive adder being responsive to the contents of the (2(M+P)-1)th and 2(M+P)th bits, respectively, of said accumulator and generating a partial sum;

the least significant M stages of each successive adder being responsive to the contents of corresponding bits of sail multiplicand register, to the M most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the contents of the M most significant stages of the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire M+2 bits of the Pth partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the (P-1)th partial sum;

whereby after the successive generation of said first and second control signals, the product of said first and second binary numbers is stored in said accumulator register.

2. The digital multiplier/accumulator circuit as recited in claim 1 and further comprising:

an array of P sign/carry circuits, one being associated with each of said full adders, each sign/carry circuit generating a sign signal and a carry signal, the first sign/carry circuit being responsive to sign and carry signals provided by said accumulator, each successive sign/carry circuit being responsive to a sign signal provided by the previous sign/carry circuit and to a carry signal provided by the most significant stage of the previous full adder.

3. A digital multiplier/accumulator circuit for generating the product of first and second binary numbers, said binary numbers each having 16 or fewer bits, said circuit comprising:

a 16-bit multiplicand register for storing said first binary number;

a 16-bit multiplier register for storing said second binary number, said multiplier register being divided into first and second 8-bit portions;

control logic for controlling the operation of said multiplier circuit, said control logic generating at least one control signal in response to one of a plurality of commands applied thereto;

an array of 4 multiplier decoders for decoding a multiplier operand which is stored in either said first portion or said second portion of said multiplier register, and for generating either a first plurality of multiplier decoder outputs or a second plurality of multiplier decoder outputs in response to first and second control signals, respectively, generated successively by said control logic;

the first of said decoders decoding the 2 least significant bits of said multiplier operand and generating 1 of 4 possible first decoder outputs representing multiplication of said multiplicand by the factors 0, +1, -1, or 2, respectively;

a second of said decoders decoding the next 2 higher significant bits of said multiplier operand, and generating 1 of 4 possible second decoder outputs representing multiplication of said multiplicand by the factors 0, +2, -2, or 4, respectively;

each successive decoder decoding the next 2 higher significant bits of said multiplier operand, with the 4th of said decoders decoding the 2 most significant bits of said multiplier operand, the Pth of said decoders generating 1 of 4 possible fourth decoder outputs representing multiplication of said multiplicand by the factors 0, +2.sup.2(P-1), -2.sup.2(P-1), or 2.sup.2P-1, respectively, where P represents the Pth decoder;

an accumulator register having 40 bit positions for storing results of the operations of said digital multiplier/accumulator circuit;

an array of 4 full adders, each being 18 bits in length,

said control logic being responsive to said first control signal for causing the least significant 16 stages of said first adder to be responsive both to the contents of corresponding bits of said multiplicand register and to the output of said first multiplier decoder, and to generate a partial sum equal to the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the two least significant bit positions of said accumulator;

the least significant 16 stages of each successive adder being responsive to the contents of corresponding bits of said multiplicand register, to the 16 most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the contents of the 16 most significant stages of the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire 18 bits of the fourth partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the third partial sum;

said control logic being responsive to said second control signal for conducting the 17th through 34th bits of said accumulator to the 1st through 18th stages, respectively, of said first adder, and enabling the least significant 16 stages of said first adder to be responsive to the contents of corresponding bits of said multiplicand register, to the contents of said 17th through 32nd bits of said accumulator, and to said first multiplier decoder generating one of said second plurality of multiplier decoder outputs, and to thereby generate a partial sum equal to the sum of said 17th through 32nd bits of said accumulator and the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the 17th and 18th bit positions of said accumulator;

the 17th and 18th stages of said first adder being responsive to the contents of the 33rd and 34th bits, respectively, of said accumulator and generating a partial sum;

the two most significant stages of the Pth successive adder being responsive to the contents of the (2(M+P)-1)th and 2(M+P)th bits, respectively, of said accumulator and generating a partial sum;

the least significant 16 stages of each successive adder being responsive to the contents of corresponding bits of said multiplicand register, to the 16 most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the 16 most significant bits of the partial product generated by the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire 18 bits of the 4th partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the 3rd partial sum;

whereby after the successive generation of said first and second control signals, the product of said first and second binary numbers is stored in said accumulator register.

4. The digital multiplier/accumulator circuit as recited in claim 3 and further comprising:

an array of 4 sign/carry circuits, one being associated with each of said full adders, each sign/carry circuit generating a sign signal and a carry signal, the first sign/carry circuit being responsive to sign and carry signals provided by said accumulator, each successive sign/carry circuit being responsive to a sign signal provided by the previous sign/carry circuit and to a carry signal provided by the most significant stage of the previous full adder.

5. A method for multiplying first and second binary numbers, said method comprising:

providing an M-bit multiplicand register for storing said first binary number, said first binary number having M of fewer bits;

providing an N-bit multiplier register for storing said second binary number, said second binary number having N or fewer bits, said multiplier register being divided into first and second equal portions;

providing an array of P multiplier decoders, where P=N/4;

employing said array of multiplier decoders to decode a multiplier operand which is stored in said first portion of said multiplier register, and generating a first plurality of multiplier decoder outputs,

the first of said decoders decoding the 2 least significant bits of said multiplier operand and generating 1 of 4 possible first decoder outputs representing multiplication of said multiplicand by the factors 0, +1, -1, or 2, respectively;

a second of said decoders decoding the next 2 higher significant bits of said multiplier operand, and generating 1 of 4 possible second decoder outputs representing multiplication of said multiplicand by the factors 0, +2, -2, or 4, respectively;

each successive decoder decoding the next 2 higher significant bits of said multiplier operand, with the Pth of said decoders decoding the 2 most significant bits of said multiplier operand and generating 1 of 4 possible Pth decoder outputs representing multiplication of said multiplicand by the factors 0, +2.sup.2(P-1), -2.sup.2(P-1), or 2.sup.2P-1, respectively, where P represents the Pth decoder;

providing an accumulator register having 2(M+P) bit positions for storing results of the operations of said digital multiplier/accumulator circuit;

providing an array of P full adders, each being (M+2) bits in length;

causing the least significant M stages of said first adder to be responsive both to the contents of corresponding bits of said multiplicand register and to the output of said first multiplier decoder, and to generate a partial sum equal to the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the two least significant bit positions of said accumulator;

the least significant M stages of each successive adder being responsive to the contents of corresponding bits of said multiplicand register, to the M most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the contents of the M most significant stages of the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire (M+2) bits of the Pth partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the (P-1)th partial sum;

conducting the (M+1)th through (2M+2)th bits of said accumulator to the 1st through (M+2)th stages, respectively, of said first adder, and enabling the least significant M stages of said first adder to be responsive to the contents of corresponding bits of said multiplicand register, to the contents of said (M+1)th through (2M)th bits of said accumulator, and to said first multiplier decoder generating one of said second plurality of multiplier decoder outputs, and to thereby generate a partial sum equal to the sum of said (M+1)th through (2M)th bits of said accumulator and the product of said first binary number and said first multiplier decoder output, the two least significant bits of said partial sum being stored in the (M+1)th and (M+2)th bit positions of said accumulator;

the (M+1)th and (M+2)th stages of said first adder being responsive to the contents of the (2M+1)th and (2m+2)th bits, respectively, of said accumulator and generating a partial sum;

the two most significant stages of the Pth successive adder being responsive to the contents of the (2(M+P)-1)th and 2(M+P)th bits, respectively, of said accumulator and generating a partial sum;

the least significant M stages of each successive adder being responsive to the contents of corresponding bits of said multiplicand register, to the M most significant bits of the partial product generated by the previous adder, and to the corresponding successive multiplier decoder output, and each generating a partial sum equal to the sum of the contents of the M most significant stages of the previous adder and the product of said first binary number multiplied by said decoder output, the two least significant bits of each successive partial sum being stored in successively higher significant bit positions of said accumulator, the entire M+2 bits of the Pth partial sum also being stored in bit positions of said accumulator which are adjacent to and successively higher than said two least significant bits of the (P-1)th partial sum;

whereby after two successive multiplications of said first binary number by said first and second portions of said second binary number, respectively, the product of said first and second binary numbers is stored in said accumulator register.

Refine Search

Search Results -

Term	Documents
(11 NOT 9).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	16
(L11 NOT L9).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	16

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L12

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Friday, February 13, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L12	L11 not l9	16	L12
L11	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near20 shift$3	25	L11
L10	L9 not l6	0	L10
L9	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near15 shift$3 near9 (left or right or most or least)	9	L9
L8	L7 and shift$3	3	L8
L7	L6 not l5	3	L7
L6	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near15 (divid$3 or division$1 or divisor$1 or dividend) near15 shift$3 near5 (left or right or most or least)	9	L6

L5	(2$2 or two$2) near5 complement$3 near15 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	6	L5
DB=PGPB,USPT; PLUR=YES; OP=OR			
L4	L3	2	L4
DB=USPT; PLUR=YES; OP=OR			
L3	(2$2 or two$2) near5 complement$3 near8 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	2	L3
L2	(2$2 or two$2) near5 complement$3 near5 (multiplication$ or multipl$5) and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	2	L2
L1	(2$2 or two$2) near5 complement$3 and normaliz$7 near8 (divid$3 or division$1 or divisor$1 or dividend) near9 shift$3 near5 (left or right or most or least)	18	L1

END OF SEARCH HISTORY

101-117

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 16 of 16 returned.

□ 1. Document ID: US 20010056453 A1

Using default format because multiple data bases are involved.

```
   L12: Entry 1 of 16                    File: PGPB                 Dec 27, 2001

PGPUB-DOCUMENT-NUMBER: 20010056453
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010056453 A1

TITLE: System and method for floating-point computation

PUBLICATION-DATE: December 27, 2001

INVENTOR-INFORMATION:
NAME                           CITY          STATE    COUNTRY      RULE-47
Steele, Guy L. JR.             Lexington     MA       US

US-CL-CURRENT: 708/497; 708/204
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 2. Document ID: US 6460063 B1

```
   L12: Entry 2 of 16                 File: USPT               Oct 1, 2002

US-PAT-NO: 6460063
DOCUMENT-IDENTIFIER: US 6460063 B1

TITLE: Division circuit and graphic display processing apparatus
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

□ 3. Document ID: US 6356927 B2

```
   L12: Entry 3 of 16                File: USPT               Mar 12, 2002

US-PAT-NO: 6356927
DOCUMENT-IDENTIFIER: US 6356927 B2
** See image for Certificate of Correction **
```

0/97153

```
TITLE: System and method for floating-point computation
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 4. Document ID: US 6327604 B1

L12: Entry 4 of 16 File: USPT Dec 4, 2001

US-PAT-NO: 6327604
DOCUMENT-IDENTIFIER: US 6327604 B1
**** See image for Certificate of Correction ****

TITLE: System and method for floating-point computation

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 5. Document ID: US 6289365 B1

L12: Entry 5 of 16 File: USPT Sep 11, 2001

US-PAT-NO: 6289365
DOCUMENT-IDENTIFIER: US 6289365 B1

TITLE: System and method for floating-point computation

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☑ 6. Document ID: US 6009451 A

L12: Entry 6 of 16 File: USPT Dec 28, 1999

US-PAT-NO: 6009451
DOCUMENT-IDENTIFIER: US 6009451 A

TITLE: Method for generating barrel shifter result flags directly from input data

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 7. Document ID: US 5928316 A

L12: Entry 7 of 16 File: USPT Jul 27, 1999

US-PAT-NO: 5928316
DOCUMENT-IDENTIFIER: US 5928316 A

TITLE: Fused floating-point multiply-and-accumulate unit with carry correction

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 8. Document ID: US 5796644 A

L12: Entry 8 of 16 File: USPT Aug 18, 1998

US-PAT-NO: 5796644
DOCUMENT-IDENTIFIER: US 5796644 A
**** See image for Certificate of Correction ****

TITLE: Floating-point multiply-and-accumulate unit with classes for alignment and normalization

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 9. Document ID: US 5787024 A

L12: Entry 9 of 16 File: USPT Jul 28, 1998

US-PAT-NO: 5787024
DOCUMENT-IDENTIFIER: US 5787024 A

TITLE: Horizontal filter in moving picture decoding apparatus

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 10. Document ID: US 5477858 A

L12: Entry 10 of 16 File: USPT Dec 26, 1995

US-PAT-NO: 5477858
DOCUMENT-IDENTIFIER: US 5477858 A

TITLE: Ultrasound blood flow/tissue imaging system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☑ 11. Document ID: US 5343416 A

L12: Entry 11 of 16 File: USPT Aug 30, 1994

US-PAT-NO: 5343416
DOCUMENT-IDENTIFIER: US 5343416 A
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for re-configuring a partial product reduction tree

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 12. Document ID: US 5065352 A

L12: Entry 12 of 16 File: USPT Nov 12, 1991

US-PAT-NO: 5065352
DOCUMENT-IDENTIFIER: US 5065352 A

TITLE: Divide apparatus employing multiplier with overlapped partial quotients

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 13. Document ID: US 4594680 A
L12: Entry 13 of 16 File: USPT Jun 10, 1986

US-PAT-NO: 4594680
DOCUMENT-IDENTIFIER: US 4594680 A

TITLE: Apparatus for performing quadratic convergence division in a large data processing system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 14. Document ID: US 3777132 A
L12: Entry 14 of 16 File: USPT Dec 4, 1973

US-PAT-NO: 3777132
DOCUMENT-IDENTIFIER: US 3777132 A
**** See image for Certificate of Correction ****

TITLE: METHOD AND APPARATUS FOR OBTAINING THE RECIPROCAL OF A NUMBER AND THE QUOTIENT OF TWO NUMBERS

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 15. Document ID: US 3626427 A
L12: Entry 15 of 16 File: USPT Dec 7, 1971

US-PAT-NO: 3626427
DOCUMENT-IDENTIFIER: US 3626427 A

TITLE: LARGE-SCALE DATA PROCESSING SYSTEM

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

□ 16. Document ID: JP 08115206 A
L12: Entry 16 of 16 File: JPAB May 7, 1996

PUB-NO: JP408115206A
DOCUMENT-IDENTIFIER: JP 08115206 A
TITLE: FLOATING POINT ARITHMETIC UNIT

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
(11 NOT 9).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	16
(L11 NOT L9).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	16

Display Format: - Change Format

Previous Page   Next Page   Go to Doc#